การเผชิญหน้าของสองผู้ยิ่งใหญ่

Quick Pascal VS Turbo Pascal

การพัฒนาภาษาปาสคาล
ที่ทวีความเข้มข้นมากขึ้นเรื่อย ๆ
ระหว่างสองบริษัทใหญ่...
ไมโครซอฟต์และ
บอร์แลนด์

เอนก ศิรจริยาภิวัฒน์

ในตอนที่ไมโครคอมพิวเตอร์เริ่มกำเนิดและเริ่มมีบทบาทในวงการธุรกิจและคอมพิวเตอร์นั้น ภาษาที่ใช้สั่งงานที่นิยมใช้และเรียกได้ว่าติดมากับไมโครคอมพิวเตอร์แทบทุกเครื่องคือภาษาเบสิก เมื่อไมโครซอฟต์จัดทำตัวแปลภาษาเบสิกขึ้นมา ทุกคนก็หันมาสนใจและใช้ Microsoft Basic ในการเขียนโปรแกรมสั่งงาน แต่ด้วยลักษณะของภาษาที่ไม่เป็นโครงสร้างของเบสิก จึงทำให้การเขียนและแก้ไขโปรแกรมเป็นไปได้ยาก จึงมีคนหันมาใช้ภาษาที่มีลักษณะโครงสร้างมากขึ้น แต่สำหรับมือใหม่ที่พึ่งเริ่มหัดเขียนโปรแกรมก็ต้องพบกับความเบื่อหน่ายเมื่อตัวเองเขียนโปรแกรมผิด แก้ไขเสร็จต้องการทดสอบก็ต้องทำการแปล (compile) กันใหม่ ซึ่งเสียเวลามากพอควร จึงทำให้มือใหม่ ๆ หลายคนหันกลับไปใช้ภาษาเบสิกเช่นเดิม เมื่อบริษัทเล็ก ๆ บริษัทหนึ่งก่อตั้งขึ้นมาในนาม Borland พร้อมกับคำว่า TURBO ตัวแปลภาษาตัวแรกที่บริษัทนี้นำออกมาสู่ท้องตลาดก็คือ Turbo Pascal ซึ่งใช้เวลาในการแปลไม่นานเหมือนกับตัวแปลภาษารุ่นเก่า ๆ บวกกับความเป็นภาษาโครงสร้างของภาษาปาสคาลเอง ทำให้ภาษาปาสคาลได้รับความนิยมมากขึ้นอย่างเห็นได้ชัด หลายคนแทบจะไม่รู้จักหรือลืมตัวแปลภาษาปาสคาลรุ่นก่อน ๆ เช่น USCD PASCAL, MT PASCAL ... ฯลฯ เมื่อได้สัมผัสกับ Turbo Pascal เทอร์โบปาสคาลครองตลาดตัวแปลภาษาปาสคาลมาตลอดจนเวอร์ชัน 5.0 ทางบริษัทไมโครซอฟต์ซึ่งเป็นผู้ครองตลาดตัวแปลภาษามาก่อน ก็เริ่มขยับตัวเพื่อต่อสู้กับตัวแปลภาษาปาสคาลอีกครั้ง (หลังจากออก Quick C มาปะทะกับ Turbo Basic) เพราะรู้สึกว่าตัวแปลภาษาปาสคาลที่มีอยู่เดิม (ซึ่งทำงานช้า) จะครองตลาดสู้ Turbo Pascal ไม่ได้ ตัวแปลภาษาปาสคาลที่ไมโครซอฟต์นำออกมาสู้กับ

ตารางที่ 1 เปรียบเทียบความสามารถของ Turbo Pascal และ Quick Pascal

Object-oriented extensions	**Quick Pascal 1.0**	**Turbo Pascal 5.5**
Dynamic objects	Yes	Yes
Stratic objects	No	Yes
Virtual methods	Yes	Yes
Static methods	No	Yes
Constructors	No	Yes
Destructors	No	Yes
Multiple inheritance	No	No
Compatibility		
Pascal	**Turbo Pascal 5.0**	**Turbo Pascal 5.0**
object extensions	**Object Pascal**	**None**
Code generation and debugging		
.TPUs	Yes	Yes
.OBJs	No	No
Smart linker	No	Yes
Integrated debugger	Yes	Yes
Add-on professional debugger	No	Yes
Documentation and help		
On-line help	Yes	Yes
Access help externally	No	Yes
Cut-and-paste examples	Yes	Yes

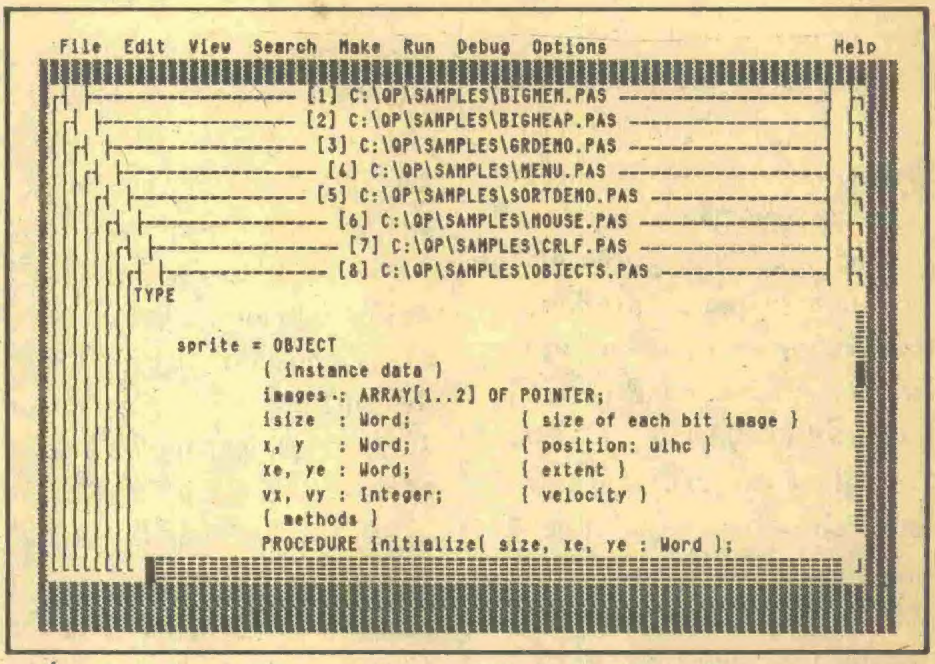
Turbo Pascal ก็คือ Quick Pascal เมื่อลูกชายคนใหม่ของตระกูล Quick จะออกมาสู้กับตระกูล Turbo จึงจำเป็นต้องติดอาวุธใหม่เข้าไป เพื่อจะได้สู้กับตระกูล Turbo ผู้แข็งแกร่งได้ เมื่อติดอาวุธใหม่แล้วก็ต้องประกาศให้ชาวโลกรู้ว่าอาวุธใหม่นั้นคือ Object Oriented Pascal ซึ่งเป็นเจ้าแรกและเจ้าเดียวบนเครื่อง IBM PC แต่ทางไมโครซอฟต์เองแทบน้ำตาร่วงเมื่อทางบอร์แลนด์ออก Turbo Pascal 5.5 ซึ่งมี Object Oriented เหมือนกัน (แต่ต่างกันที่การนำเสนอเล็กน้อย) เราก็คงต้องมาดูกันต่อไปว่าแต่ละฝ่ายมีความแข็งแกร่งอย่างไร และสงครามระหว่างตระกูล Quick กับตระกูล Turbo จะเป็นอย่างไร

QUICK PASCAL

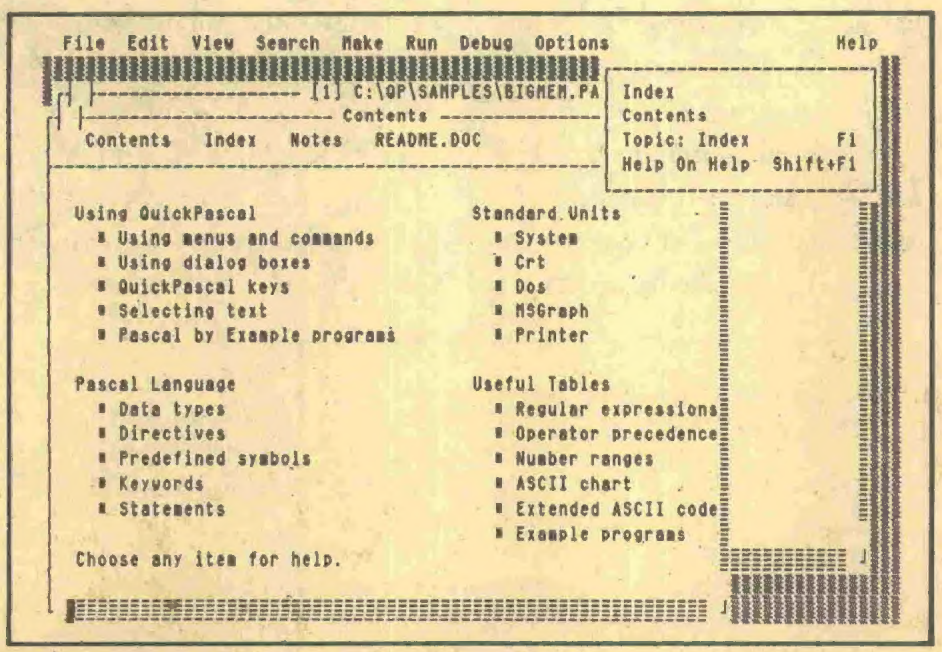
Turbo Pascal ครองตลาดตัวแปลภาษาปาสคาลมานาน จึงเรียกได้ว่าเป็นมาตรฐานอีกมาตรฐานหนึ่ง เมื่อไมโครซอฟต์ออก Quick Pascal จึงทำให้เข้ากันได้ (compatible) กับ Turbo Pascal version 5.0 และเพิ่มความสามารถของ Object Oriented Programming (OOP) เข้าไป แต่ Turbo Pascal ก็มีให้จึงต้องมาดูแง่การใช้งานและลักษณะพิเศษอื่น ๆ ประกอบเข้าไปด้วย เราเริ่มมาดูในสิ่งที่ไมโครซอฟต์บอกว่า Quick Pascal ของตัวเองมีกันก่อน

- มีการแปลภาษา (compile) แบบ command line และ integrated ในแบบ integrated นั้นสามารถเปิดเอดิเตอร์พร้อมกันได้ 9 เอดิเตอร์ สามารถเคลื่อนย้าย เปลี่ยนขนาดของเอดิเตอร์แต่ละตัวได้แยกอิสระต่อกันดังรูปที่ 1

- มีระบบความช่วยเหลือที่ดี ขนาดไม่ต้องมีคู่มือเป็นเอกสารอ้างอิงและนอกจากจะมีตัวอย่างโปรแกรมในแผ่นดิสค์แล้วยังมีตัวอย่างในการขอความช่วยเหลือให้อีก โดยการเลือก Pascal by Example Programs ในเมนู content (รูปที่ 2) ซึ่งสามารถคัดลอกลงเอดิเตอร์ แล้วทดลองให้โปรแกรมทำงาน



รูปที่ 1 Quick Pascal สามารถใช้เอดิเตอร์ได้พร้อมกันถึง 9 อัน



รูปที่ 2 ระบบการขอความช่วยเหลือของ Quick Pascal

ได้เป็นอย่างดีและสำหรับท่านที่กลัวว่าจะใช้ integrated environment ของ Quick Pascal ไม่เป็น ก็มีโปรแกรม LEARN.EXE ที่พาท่านทำงานกับ integrated environment ของ Quick Pascal และท่านพอใจจะเรียนแค่ไหนท่านก็สามารถหยุดเพียงเท่านั้นโดยโปรแกรม LEARN จะเก็บขั้นตอนที่ท่านเรียนไปแล้วไว้ให้

- กราฟิกตรงกัน (Compatible) กับ Quick C 2.0 และ Borland Graphics Interface (BFI)

- มี Object Oriented Programming ให้

- มีระบบการแก้ไขและหาข้อผิด

พลาด (debug) ที่ดี

- ลักษณะการสร้างภาษา (Syntax) ตรงกับ Turbo Pascal 4.0 และ 5.0 ซึ่งสามารถทำการแปลโปรแกรมเดิมที่เขียนด้วย Turbo Pascal โดยแทบไม่ต้องแก้ไข

เนื่องจากระบบความช่วยเหลือที่มีให้ใน integrated environment ดีมาก จึงมีคู่มือเพียง 2 เล่มคือ Up and Running เป็นการบอกการติดตั้ง Quick Pascal และ Pascal by Example เป็นเล่มที่สอนการใช้งานภาษาปาสคาล เล่มที่น่าจะมีแต่ไม่มีก็คือ Programmer Reference Guide ซึ่งเอกสารอ้างอิงอยู่ในระบบการช่วยเหลือทั้งหมด บางคนก็ว่าดีเพราะไม่ต้องเสียเวลาเปิดหนังสือ บางคนก็ว่าไม่ดีเพราะจะอ่านที่อื่นที่ไม่ใช่หน้าคอมพิวเตอร์ก็ไม่มีให้ ทำให้อึดอัดพอสมควร ที่สำคัญก็คือสำหรับการใช้เอดิเตอร์สำหรับผู้เริ่มต้น จะไม่มีบอกให้ในคู่มือ การจะดูว่าจะใช้เอดิเตอร์อย่างไรต้องดูผ่าน on line help ซึ่งจำเป็นต้องทำถึง 3 ขั้นตอน (ต่างกับ Turbo Pascal ที่กด F1 เพียงขั้นตอนเดียว) แต่เอดิเตอร์ของ Quick Pascal ก็สนับสนุน mouse ทำให้ใช้งานง่ายมากขึ้น ผู้ที่เป็นโปรแกรมเมอร์และเคยใช้เอดิเตอร์ตัวดัง ๆ ตัวอื่นที่ไม่ใช่รูปแบบ WORD STAR (เอดิเตอร์ที่ Quick Pascal ตั้งให้) ก็สามารถดิลตั้งเป็นคีย์ที่ตัวเองถนัด ซึ่งเอดิเตอร์ที่ Quick Pascal สนับสนุนก็ดูได้จากรูปที่ 3 และรูปแบบการใช้เอดิเตอร์ใหม่ที่ Quick Pascal นำเสนอก็คือ Pascal Syntax ซึ่งจะบอกว่าคำที่เราคีย์เป็นส่วนใดของ Pascal Syntax โดยแสดงออกมาหลาย ๆ สี สำหรับสีที่ติดตั้งครั้งแรกคือ keywords จะเป็นสีม่วง comment จะเป็นสีเขียว ข้อความในเครื่องหมายคำพูดจะเป็นสีฟ้า นอกจากที่กล่าวมาจะเป็นสีขาว สำหรับการปฏิบัติงาน (RUN) ของโปรแกรมที่เราเขียน เมื่อปฏิบัติงานเสร็จจะไม่ถูกทับซ้อนเหมือน Turbo Pascal (ต้องกด Alt-F5 ถึงจะเห็น) โดยจะรอให้คนเขียนโปรแกรมเคาะคีย์ใด ๆ ก่อน พร้อมด้วยการบอกเวลาที่ใช้ในการปฏิบัติงาน แต่สิ่งหนึ่งที่ Quick Pascal ขาดไปก็คือการทำ Overlay จึงไม่สามารถเขียนโปรแกรมใหญ่กว่าขนาดหน่วยความจำของเครื่องที่มีอยู่ได้ ซึ่งอาจเป็นปัญหาที่จะใช้ Quick Pascal ผลิตซอฟต์แวร์ขนาดใหญ่ออกจำหน่าย สำหรับยูนิตนั้น Quick Pascal ใช้ชื่อเหมือนกับ Turbo Pascal ยกเว้นกราฟิกซึ่งใช้ MsGraph แทน Graph ยูนิตของ Turbo Pascal ซึ่ง Quick Pascal บอกว่ากราฟิกของตน Compatible กับกราฟิกของ Turbo Pascal 4.0 & 5.0 แต่ผู้เขียนลองใช้กับโปรแกรมที่เขียนขึ้นมาด้วย Turbo Pascal 5.0 แล้วปรากฏว่าคอมพายล์ผ่านโดยไม่ต้องเปลี่ยนยูนิต Graph เป็น MsGraph แต่ปรากฏว่าเมื่อทดลองให้โปรแกรมปฏิบัติงาน (RUN) แล้วพบว่ารูปกราฟิกที่เขียนไว้ไม่แสดงผล จึงเปลี่ยนยูนิต Graph เป็น MsGraph แล้วคอมพายล์ใหม่ ปรากฏว่าคอมพายล์ไม่ผ่านติดที่คำสั่ง SetTextjustify ซึ่งแสดงว่าไม่ compatible กับ Turbo Pascal 5.0 อย่าง 100% อีกเรื่องหนึ่งที่ลองใช้แล้วรู้สึกอึดอัดพอสมควรคือถ้าไม่มี mouse ขอความช่วยเหลือแล้วต้องการปิดหน้าต่างขอความช่วยเหลือถ้าเป็นการขอความช่วยเหลือเกี่ยวกับเมนูจะใช้คีย์ Esc ถ้าเป็นการขอความช่วยเหลือในเอดิเตอร์จะปิดโดย CTRL-F4 ซึ่งเป็นคีย์ที่ต่างกันจึงทำให้ใช้งานพลาดเสมอและเมนู change directory ในตระกูล Quick ก็ไม่มีตั้งแต่เริ่มตระกูล จึงทำให้ไม่สะดวกนัก

TURBO PASCAL 5.5

ในเวอร์ชัน 5.5 แทบจะไม่มีอะไรเปลี่ยนแปลงจากเวอร์ชัน 5.0 นอกจากการทำ Overlay ที่ดีขึ้นและ Object Oriented Programing แต่การเปิดตัวของเวอร์ชัน 5.5 นี้ทำให้ทางบอร์แลนต้องออกซอฟต์แวร์เวอร์ชันใหม่ตามมาด้วยนั้นคือ Turbo Assembler 1.5 เพื่อสนับสนุนการ debug แบบ stand alone ของ Object Oriented Programming และทางไมโครได้แนะนำ Turbo Pascal 5.5 ไปแล้ว จึงไม่ขอแนะนำอีก

Oject Oriented Programming (OOP)

Object Oriented Programming เกิดขึ้นตั้งแต่ ค.ศ. 1960 มีบริษัทหลายบริษัทพยายามพัฒนาระบบ OOP นี้ แต่ที่สำเร็จและเป็นตัวเป็นตนขึ้นมาก็คือโครงการ Palo Alto Research Center (PARC) ของบริษัท Xerox ที่ใช้เวลา 10 ปีพัฒนา Smalltalk ขึ้นมาซึ่งเป็น OOP ตัวแรกในวงการ สำหรับ Smalltalk/V (ของบริษัท Degitalk) และ OOP เบื้องต้น ทางไมโครได้นำเสนอกับท่านผู้อ่านแล้ว บทความนี้จึงเน้นเป็น OOP ของภาษา

```
 Volume in drive C is ANEK
 Directory of  C:\OP

BRIEF    KEY      273    5-04-89   5:03a
EMACS    KEY      303    5-04-89   5:04a
EPSILON  KEY      303    5-04-89   5:04a
ME       KEY      233    5-04-89   5:03a
OP       KEY      533    5-04-89   5:03a
       5 File(s)  16807936 bytes free
```

รูปที่ 3
คีย์ในเอดิเตอร์ที่ Quick Pascal สนับสนุนการใช้งาน

ปาสคาล ซึ่งเดิมภาษาปาสคาลไม่มีลักษณะของ OOP แต่ถูกทำให้เป็น OOP ของภาษาปาสคาล ซึ่งเดิมภาษาปาสคาลไม่มีลักษณะของ OOP แต่ถูกทำให้เป็น OOP โดยบริษัท Apple ในปี 1985 ซึ่ง Quick Pascal จะยึดหลักของ OOP ของ Apple Pascal นี้แต่ Turbo Pascal จะรวมความสามารถของ C++ ลงไปด้วย ลักษณะของ OOP ในปาสคาลจะเป็นรูปแบบของเรคอร์ด เพราะข้อมูลแบบเรคอร์ดนี้มีความสามารถที่จะรวมสิ่งต่าง ๆ เข้าด้วยกัน เช่น เราจะใช้ข้อมูลของยานอวกาศ เราก็สามารถทำเป็นเรคอร์ดได้ดังนี้

```
type
   SpaceShip = record
      Length, Shields, Photon_torps,
      Phaser_strength: Integer;
      Disabled : boolean;
   end;
```

เมื่อเรากำหนดรูปแบบของยานอวกาศเป็นเรคอร์ดแล้ว เราก็จะถือว่าคุณสมบัติของยานอวกาศมีเพียงหนึ่งเพราะถ้าเรากำหนดฝ่ายดีและฝ่ายร้ายโดย

```
Var Goodguys, Badguys : Spaceship;
```

เราก็สามารถเปรียบเทียบยานอวกาศทั้งสองชนิดได้เช่น

```
if Goodguys = Badguys then ...
```

นอกจากนี้ยังใช้คุณสมบัติของยานอวกาศแต่ละลำได้โดยการใช้เครื่องหมาย .(dot) เช่น

```
if Goodguys.Disabled then ...
```

เราจะสังเกตว่าข้อมูลแบบ record เป็นข้อมูลแบบโครงสร้าง เมื่อ OOP เข้ามาสู่ปาสคาล OOP จึงใช้ข้อมูลแบบ record เพราะ record บอกคุณสมบัติต่าง ๆ ได้ดีนั้นเอง และเราจะเรียกใหม่ว่าแบบ object เพราะข้อมูลแบบ record มีแต่ข้อมูล ไม่มีการกระทำกับข้อมูลนั้นเอง นอกจากตัว object แล้วยังมีการกระทำกับ object ซึ่งเรียกว่า method นอกจากนั้น object ยังสามารถมีลูกหลานได้ (inherit) โดย object ลูกหลานนี้จะมีคุณสมบัติของบรรพบุรุษของมันเอง สมมติว่า object ของยานอวกาศเป็นต้นกำเนิด (parent) ก็จะมีลูก (child object) ได้เป็นยานบรรทุกและยานโจมตีดังรูปที่ 4 โดยยานบรรทุกและยานโจมตีจะมีคุณสมบัติเหมือนยานอวกาศ (บรรพบุรุษ) และมีคุณสมบัติของตัวเองเพิ่มขึ้นมา ซึ่งเราจะใช้ลักษณะเขียนโปรแกรมได้ดังนี้

```
type
   Ship = object
      X, Y, Shield : Integer;
      Side : Char;
   end;
      {Cargo inherits Ship attributes }
      Cargo = object(Ship)
   Max_weight, Cargo_weight: Integer;
   end;
   Attack = object(Ship)
      Photons:Integer;
```

เมื่อเรากำหนด object แบบ Cargo และ Attack แล้วเราก็สามารถใช้ object นั้นได้ เช่น

```
Var
      Myfighter, Yourfighter : Attack;
      Myfreighter, Yourfreighter : Cargo;
```

เมื่อเรากำหนดตัวแปรเป็นแบบ object เราก็สามารถเข้าถึงคุณสมบัติของ object นั้นได้ด้วยวิธีของ record เช่น เราจะเข้าถึง Myfreighter ก็จะมี Myfreighter.X, Myfreighter.Y, Myfreighter.Shield, Myfreighter.Side, Myfreighter.Max_weight และ Myfreighter.Cargo_weight ซึ่งที่เราดูผ่านกันมานี้มีเฉพาะตัว object เรารู้แล้วนอกจากจะมีตัว object แล้วยังมีวิธีการ (method) ที่กระทำกับ object นั้นได้ด้วย ซึ่งวิธีการนี้เราจะเขียนไว้ในการประกาศ object เลย ซึ่งเราจะเห็นว่า object นี้ไม่เหมือน Record เพราะ Record ไม่สามารถมีวิธีการใน Record นั้นได้และไม่สามารถมีลูกหลานได้ สมมติว่าเราต้องการให้ยานอวกาศอยู่ที่ตำแหน่ง X ที่เราต้องการ เราก็จะกำหนด object ใหม่ได้เป็น

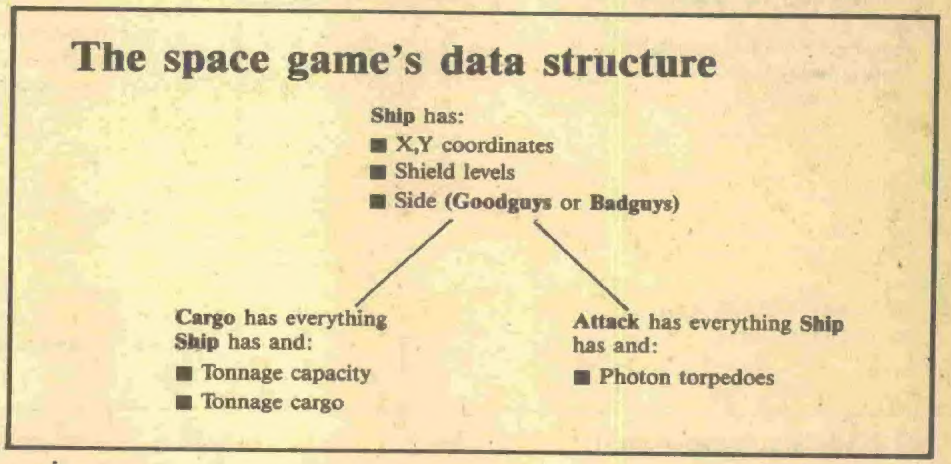
```
type
      Ship = object
         X, Y, Shield : Integer;
         Side         : Char;
         procedure InitX(NewX : Integer);
end;
```

ส่วนวิธีการนั้นเราเขียนแยกการกระทำทั้งหมดออกมาจาก object (เพียงบอกการกระทำในตัวแปร object แต่ขั้นตอนเขียนแยกออกมาต่างหาก) ซึ่งวิธีการนี้ก็คือ หลักภาษาปาสคาลธรรมดานั่นเอง ซึ่งเราจะเขียนวิธีการ InitX ได้เป็น

```
procedure InitX(NewX : Integer);
begin
      X := NewX ;
end;
```

Turbo Pascal OOP

การใช้ object นั้นมี 2 ลักษณะ คือ ไม่มีการเปลี่ยนแปลง object หมาย-



รูปที่ 4 ลูกหลานของยานอวกาศ (ship)

ถึง ถ้า object ถูกนิยามแล้ว เมื่อทำการแปล (compile) แล้ว object ไม่เปลี่ยนแปลงอีก เราจะเรียกว่า static ซึ่งเปรียบเสมือน object นั้นเป็น ชิพ (chip) IC ที่ไม่สามารถโปรแกรมได้ เราเพียงนำชิพเหล่านั้นมาต่อเป็นวงจรที่เราต้องการโดยให้สัญญาณอินพุต (method) กับชิพนั้น เราจะได้เอาต์พุตตามต้องการ การทำงานแบบ static นี้ไม่ต้องทำการตีความหมาย (interpret) ขณะโปรแกรมทำงาน เพราะฉะนั้นโปรแกรมจึงทำงานได้เร็ว อีกวิธีเรียกว่า virtual จะทำให้ object นั้นยืดหยุ่นขึ้น (open and flexible) ซึ่งเป็นการเพิ่มประสิทธิภาพการทำงานขึ้น แต่วิธีการนี้จำเป็นต้องตีความหมายขณะปฏิบัติงาน (Run time) จึงทำให้การทำงานช้าลง การใช้งานแบบ virtual นี้จะใช้คำ 2 คำ เพื่อบอกว่าเป็น virtual method นั้นคือ constructor และ virtual ตัวอย่างก็ได้แก่

```
type
   Ship = object;
      X, Y, Shield : integer;
      Side         : char;
      Constructor InitX(NewX : Integer);
end;
```

ซึ่งจะเห็นว่าแทนที่เราจะเรียกว่า procedure เราก็เรียกว่า constructor แทน ซึ่ง constructor นี้สำคัญมาก ถ้าจะใช้ Virtual Method Table (VMT) ซึ่งสร้างขึ้นมาขณะปฏิบัติงาน (Run-time) ก็หมายความว่าถ้าจะใช้ virtual method ต้องมี constructor ถ้าไม่มีก็เดาไม่ถูกว่าโปรแกรมจะเป็นอย่างไร (Twilight Zone) เพราะ constructor นี้เป็นตัวบอก compiler ให้ใช้วิธี virtual ส่วนคำว่า virtual ก็เป็นคำที่บอกวิธีการ virtual เช่นกัน ตัวอย่างเช่น เราต้องการใช้วิธีการ virtual กับการกระทำ changX เราก็จะเขียนโปรแกรมได้เป็น

```
type
   ship = object
      X, Y, Shield : Integer;
      Side         : Char;
      Constructor InitX (NewX:Integer);
      procedure ChangX (NewX:Integer);
         virtual;
end;
```

สำหรับโปรแกรมตัวอย่างของยานอวกาศ รูปแบบของ Turbo Pascal สามารถดูได้จากโปรแกรมในรูปที่ 5 และการ compile แบบ object จะ

```
program objexp;

type
  location = object
    X,Y:integer;
  procedure init(NewX,NewY:integer);
  end;

  ship = object(location)
    Shield:integer;
    Side:char;
    procedure init(NewX,NewY,NewShield:integer;NewSide:char);
  end;

  procedure location.init(NewX,NewY:integer);
  begin
    x:=NewX;
    y:=NewY;
  end;

  procedure ship.init(NewX,NewY,NewShield:integer;NewSide:char);
  begin
    location.init(NewX,NewY);
    Shield:=NewShield;
    Side:=NewSide;
  end;

var

  Ship1:Ship;

begin
  Ship1.Init(10,20,100,'W');
  writeln(ship1.side);
end.
```

รูปที่ 5 โปรแกรมยานอวกาศแบบ OOP เขียนด้วย Turbo Pascal

```
program objexp;
{M+}

type
  location = object
    X,Y:integer;
  procedure ilocation(NewX,NewY:integer);
  end;

  ship = object(location)
    Shield:integer;
    Side:char;
    procedure iship(NewX,NewY,NewShield:integer;NewSide:char);
    end;

  procedure location.ilocation(NewX,NewY:integer);
  begin
    self.x:=NewX;
    self.y:=NewY;
  end;

  procedure ship.iship(NewX,NewY,NewShield:integer;NewSide:char);
  begin
    self.ilocation(NewX,NewY);
    self.Shield:=NewShield;
    self.Side:=NewSide;
  end;

var

  Ship1:Ship;

begin
  new(Ship1);
  Ship1.Iship(10,20,100,'W');
  writeln(ship1.side,ship1.x);
end.
```

รูปที่ 6 โปรแกรมยานอวกาศแบบ OOP เขียนด้วย Quick Pascal

ช้ากว่าวิธีการอื่นจึงมีข้อแนะนำว่าให้ใช้ object ในยูนิตไม่แนะนำให้เขียน object ใน main program แล้ว compile ยูนิตที่บรรจุ object นั้นให้เสร็จ ซึ่งจะทำให้ประหยัดเวลาในการพัฒนาโปรแกรม

Quick Pascal OOP

สำหรับ OOP ของ Quick Pascal จะยึดหลักของ MacApp Object oriented ของบริษัท Apple ซึ่ง OOP ของ Quick Pascal ไม่มีวิธีการแบบ virtual และมีข้อแตกต่างกับ Turbo Pascal ดังนี้

1) Quick Pascal ต้องมี compiler directive {M+} เพื่อบอกว่าจะใช้ OOP

2) Quick Pascal ต้องการคำ Override ถ้ามีวิธีการที่ object นั้นเป็นวิธีการที่ต่างจากบรรพบุรุษ (ขอให้ดูวิธีการ Getchar ในโปรแกรมตัวอย่างซึ่งเขียนโดย Turbo Pascal ในโปรแกรมในรูปที่ 7)

3) Quick Pascal ไม่ยอมให้ใช้ชื่อวิธีการที่มีขั้นตอนต่างกัน ดังนั้นจึงใช้ Init กับ ship (มีพารามิเตอร์ 4 ตัว) และ Init กับ Location ซึ่งมีพารามิเตอร์ 2 ตัวพร้อมกันไม่ได้ ซึ่งจะเห็นได้ในวิธีการ Init ของ Turbo Pascal (โปรแกรมในรูปที่ 5) และ Quick Pascal (โปรแกรมในรูปที่ 6)

4) Quick Pascal ต้องการคำว่า New ในการสร้าง object แต่ใน Turbo Pascal จะใช้เมื่อต้องการทำ dynamic structures ซึ่งจะเห็นข้อแตกต่างนี้ได้

รูปที่ 7
โปรแกรมเปลี่ยนไฟล์ WordStar เป็นไฟล์ ASCII แบบ OOP

```
Program Ws_to_ASCII;
uses Dos;

type
    Str80 = String[80];
    ASCII = Object
                body : Text;
                function  OpenFile(FileName : Str80) : Boolean;
                function  CreateFile(FileName : Str80) : Boolean;
                function  GetChar : Char;
                procedure PutChar(Ch       : Char);
    end; { ASCII object}

    WordStar = Object(ASCII)
                function GetChar : Char;
    end; { WordStar object}

var
   InFile  : WordStar;
   OutFile : ASCII;

{-------------------------------------------------------}
function ASCII.OpenFile(FileName : Str80) : Booolean;
begin
     Assign(body,FileName);
     {$I-} Reset(body); {$I+}
     OpenFile := IOResult = 0;
end; {Open}

function ASCII.CreateFile(fileName : Str80) : Boolean;
begin
     Assign(body,FileName);
     {$I-} ReWrite(body); {$I+}
     CreateFile := IOResult = 0;
end; { Create }

function ASCII.GetChar : Char;
var
```

→

```
   Ch : Char;
begin
     Read(body,Ch);
     GetChar := Ch;
end; { Get a Char }
procedure PutChar(Ch : Char);
begin
     write(body,Ch);
end; { Put a Char }
{-------------------------------------------------------}
function WordStar.GetChar : Char;
var
   Ch : Char;
begin
     Read(body,Ch);
     if Ord(Ch) <> $1E then
        GetChar := Chr(Ord(Ch) and $7F)      { zero 8th bit }
     else
        GetChar := '-';                      { make soft hyphen a real one
}
end; { Get a Char for WordStar }

procedure Instructions;
begin
     writeln('----------------------------');
     writeln(' WordStar file conversion');
     writeln('----------------------------');
     writeln;
     writeln('WS wsfile ascfile');
     writeln;
     writeln('converts wsfile(WordStar format) to ascfile(ASCII format).');
end; { Instruction }


begin { main program }
      if paramCount < 2 then
         Instructions
      else
          begin
               if not InFile.OpenFile(ParamStr(1)) then
                  begin
                       writeln('Can''t open WordStar file');
                       halt;
                  end;
               if not OutFile.CreateFile(ParamStr(2)) then
                  begin
                       writeln('Can''t create ASCII file');
                       halt;
                  end;
               while not eof(InFile.Body) do
                     OutFile.PutChar(InFile.GetChar);
               Close(InFile.Body);
               Close(OutFile.Body);
          end;
 end.
```

■

รูปที่ 8 ทีมงาน Turbo Pascal

จาก main program ของรูปที่ 5 และรูปที่ 6

5) Quick Pascal จะอ้างถึงส่วนต่าง ๆ ของ object โดยตัวมันเอง (self) จะใช้ with ไม่ได้ (wish self do ...)

สำหรับ OOP ที่เขียนด้วย Quick Pascal ที่มีคุณสมบัติเหมือนที่เขียนด้วย Turbo Pascal (โปรแกรมในรูปที่ 5) ก็คือโปรแกรมในรูปที่ 6

ส่งท้าย

ผมได้นำโปรแกรมที่แปลงไฟล์ของ WordStar ไปเป็น ASCII ไฟล์มาฝากดังรูปที่ 7 โดย WordStar จะเก็บ hyphen (-) เป็น 1Eh เพราะฉะนั้นจึงทำเพียงเปลี่ยน 1Eh เป็นโปรแกรมที่นำมานี้เพื่อให้ดูลักษณะ OOP ได้ชัดเจนขึ้น โปรแกรมนี้เขียนด้วย Turbo Pascal โดยเราจะกำหนด ASCII เป็น object โดยตัว object เป็น Text file มีการกระทำคือเปิดไฟล์, สร้างไฟล์, นำข้อมูลจากไฟล์และใส่ข้อมูลกับไฟล์ ส่วน WordStar จะเป็นลูกของ ASCII อีกที เพียงแต่วิธีการนำข้อมูลจากไฟล์ (GetChar) ต่างกัน และนำมาฝากอีกอย่างคือรูปทีมงานของ Turbo Pascal ดังในรูปที่ 8 ซึ่งทำงานมาตั้งแต่เวอร์ชัน 1.0 จนถึง 5.5 เรียงลำดับจากซ้ายไปขวามือ Anders Hejlsberg อายุ 28 ปี, Gary Whizin อายุ 32 ปี และ Michael Weisert อายุ 29 ปี หลายคนไม่รู้ OOP ใช้งานได้ดีขึ้นแค่ไหนจะยุ่งยากขึ้นหรือเปล่า? สำหรับความนิยมของ OOP นั้นเพราะสามารถแก้ไขและตรวจสอบ (debug) โปรแกรมได้ง่าย, เมื่อมี object หนึ่งแล้วสามารถทำ Object ใหม่ได้. ทำให้สามารถนำโปรแกรมเก่ามาใช้งานได้ง่าย ฯลฯ OOP มีความสามารถมากและประยุกต์ใช้ได้ดี จะเห็นความสามารถของ OOP เมื่อเราเขียนโปรแกรมใหญ่ ๆ ที่มีความสลับซับซ้อนมาก ๆ เราลองมาฟังคำพูดของทีมงาน Turbo Pascal สักหน่อยแล้วกันว่าเขามีความคิดเห็นอย่างไรกับโอโอพี Michael Weisert กล่าวว่า "Programs have become complex, and objects offer a solution" Anders Hejlsbert กล่าวว่า "There are benefits in breaking large tasks into little, easily identifiable, maintainable chunks. There'no big source file that everyone wants every five minutes, You have little objects with well-defined interfaces that people can fine-tune. So I definetely think that when you get into multiple-programmer projects, objects are going to be big help" แต่กระนั้น Anders Hejlsberg ก็บอกว่า OOP ไม่ใช่สิ่งที่แก้ปัญหาได้หมดทุกอย่าง โดยกล่าวว่า "Some things are just not done with objects, Straight-line code for mathematic calculation in always going to be straight-line code unless you get into parallel processor or what have you. Certain kinds of issues just don't have anything to do with objects" ซึ่งคำกล่าวเหล่านี้คงทำให้เรามอง OOP ได้กว้างขึ้น ตอนนี้ก็ได้ข่าวว่าทางบอร์แลนด์จะออก Turbo C ซึ่งมีลักษณะC++ ให้, Turbo Pascal เวอร์ชันใหม่ซึ่งทั้งคู่สนับสนุนการเขียนโปรแกรมบน Window เราคงต้องมาดูทางไมโครซอฟต์จะออกอะไรมาสู้เพื่อสิทธิภาพของตัวเองเพราะไมโครซอฟต์เป็นผู้พัฒนา Window เอง ส่วนตัวผมนั้นมองอยู่ว่าใครจะออก Turbo Fortran, Turbo Cobol, Turbo modula, Turbo Ada หรือจะเป็นตระกูล Quick ก็ได้เพราะจะทำให้การศึกษาและพัฒนาซอฟต์แวร์ในเมืองไทยมีความรวดเร็วขึ้น ผมกำลังรออยู่แล้วท่านละครับ